第2章 サイドチャネル攻撃のからくりを理解する

—— DESとRSAに対する攻撃と基本的な防御法

佐藤 証

ここでは，正規の経路以外から暗号LSIの内部情報を読み取るサイドチャネル攻撃のしくみを解説する．具体例として，公開かぎ暗号のRSAと共通かぎ暗号のDES（Data Encryption Standard）を取り上げる．暗号アルゴリズムの特徴や回路の実装方法に応じて攻撃のしかたは異なる．攻撃法を理解しておけば，有効な対策を打つ際の手がかりになる．（編集部）

暗号LSIモジュールへの攻撃法にはさまざまな種類がありますが，**図1**に示すように，大きく「破壊攻撃」と「非破壊攻撃」に分類できます．

破壊攻撃は，特別な装置を備えた実験室において，LSIのパッケージを開けて行います．回路パターンを解析して回路図を起こしたり，直接メモリの内容を読み出したり（あるいはメモリに書き込んだり），さらにはLSIの中の配線をつなぎ変えるといった芸当すら可能です．しかし，こうした破壊攻撃は非常に高価な装置と高いスキルが要求されるため，個人レベルでは実行不可能です．

これに対して，非破壊攻撃は内部動作に応じて変化する電流や電圧，電磁波，処理時間などを観測します．対策が施されていない回路は，オシロスコープやパソコンといった簡単な設備で攻撃できます．サイドチャネル攻撃は，この非破壊攻撃に分類されます．

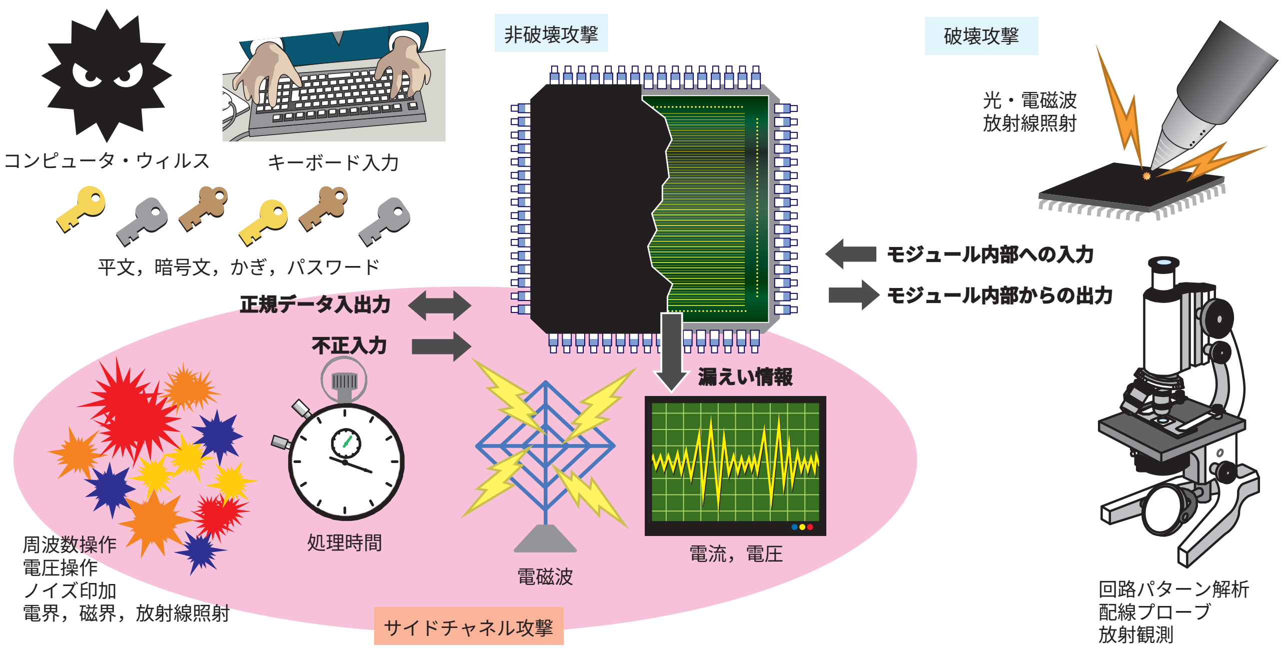


図1　暗号LSIモジュールへのさまざまな攻撃法
サイドチャネル攻撃は非破壊攻撃に分類される．これは，正規の入出力以外のチャネルを通じて観測されたデータをもとに秘密情報を解析する手法である．

● 情報を正規の経路以外から取り出す

サイドチャネルとは，正規の入出力経路ではないことを意味しています．つまり，サイドチャネル攻撃はそうした経路から観測できるあらゆる情報を使って，LSIの内部のデータを読み取ろうとするものです．ただし，キーボードやテンキーの状態を監視して，暗号LSIを始動させるためのパスワードやピンを割り出すというのは範囲外です．あくまでも，LSI内部にある秘密情報を取り出すというのが攻撃の目的です．

また，サイドチャネル攻撃では，正規のデータ入出力にほかの観測情報を組み合わせて解析を行います．このとき，統計処理が行いやすいように入力データのパターンを1ビットずつ変えたり，LSIに与える電圧や動作周波数を変えたり，あるいはスパイク・ノイズを電源へ印加したり，電子ビームを照射することによって計算を誤らせてかぎ情報の漏えいをねらうといった手法もあります．

RSA暗号を作った3人のうちのひとりであるShamirは，米国Intel社のCeleronプロセッサにRSA暗号ソフトウェアを実装し，秘密かぎごとに発生する音が異なることを利用した音響攻撃を示しています(1)．この音は，LSIの発熱によって生じます．**図2**に示すように，スプレーによる冷却という不正入力（？）で音のスペクトルが変化するようすがわかります．

サイドチャネル攻撃の研究は，公開かぎ暗号の処理時間が秘密かぎのビット・パターンによって異なることを利用する解析手法(2)を，1996年にKocherが提案したのが始まりです．それからまだ10年しかたっていませんが，解析や防御手法の研究はますます活発に行われています．

以下に，その中でもっとも基本的な二つの攻撃法を説明します．一つは単純電力解析（SPA：simple power analysis），もう一つは差分電力解析（DPA：differential power analysis）です．これらを，RSAとDESの二つの暗号に対して適用する例を挙げます．

1 RSAをSPAで攻撃する

RSAは公開かぎ暗号の代表とも言えるもので，暗号化と復号に異なるかぎを用います．xを元のデータ（平文と言う），yを暗号文，eとnを暗号化かぎ，dを復号かぎとするとき，暗号化と復号は次のような簡単なべき乗剰余算の式で表されます（$a \bmod b$は，aをbで割った余りを意味する）．

暗号化：$y = x^e \bmod n$ ……………………………(1)

復号　：$x = y^d \bmod n$ ……………………………(2)

かぎの作りかたは専門書に譲るとして，まずは簡単な例を示しましょう．平文を$x = 2$，暗号化かぎを$e = 9$，$n = 69$，復号かぎを$d = 49$とします．式(1)から暗号文は29（$= 2^9 \bmod 69$）となります．これを式(2)に当てはめると，$x = 2$（$= 29^{49} \bmod 69$）となり，平文2と暗号文29の相互変換がべき乗剰余算で確かにできていることがわかります．試しに，暗号文29を暗号化かぎで変換しても$29^9 \bmod 69 = 62$となり，平文（この場合は2）に戻すことはできません．

この暗号化かぎはだれに渡してもよいため，公開かぎ（public key）と呼ばれます．これが公開かぎ暗号の名まえの由来です．また，復号かぎは秘密かぎ（private key）と

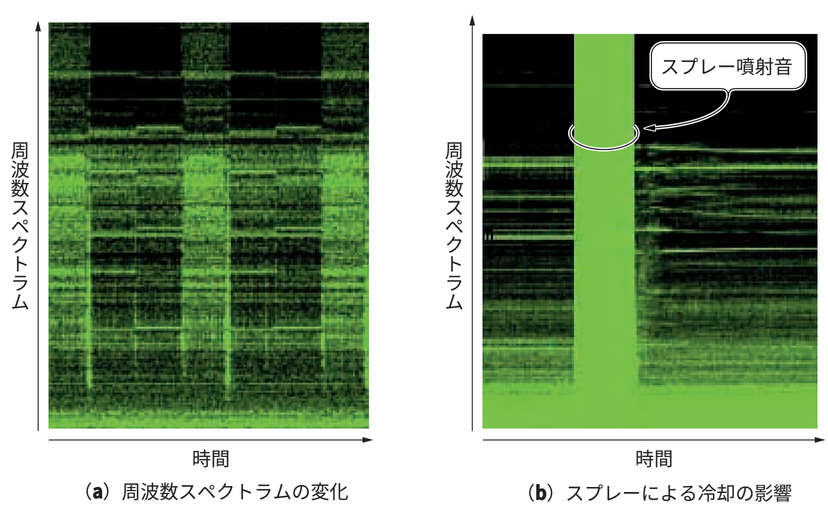


図2
Shamirの音響攻撃による観測波形
図は，RSA暗号ソフトウェアをCeleronプロセッサに実装し，動作中にプロセッサが発生する音の周波数を横軸に，時間を縦軸にとってグラフ化したもの．(a)では演算処理に応じてスペクトラムが変化するようすがわかる．プロセッサの発熱によって音が生じるため，スプレーで冷却すると，(b)のようにスペクトラムが変化する．

呼ばれ，暗号通信を行いたい人は公開かぎを相手に渡す一方，そのペアとなる秘密かぎは絶対安全に守らなくてはなりません．DESなどの共通かぎ暗号は，暗号化と復号に同一の共通かぎを用いるため，遠くの相手と初めて通信するときにその共通かぎをどうやって安全に渡すかが問題となりますが，公開かぎ暗号はこの問題を解決してくれます．

公開かぎ9で暗号化したものは秘密かぎ49でないと元に戻せませんが，9と49のペアなんてちょっと計算すれば簡単に求まるように思えます．ですが，RSAで扱う数は1,024～4,096ビットといった大きなもので，秘密の情報（$n = 69 = 3 \times 23$といった二つの因数）を知らないと秘密かぎを求めることは計算量的に不可能なのです（下掲のコラム「RSA暗号と素因数分解と離散対数問題」を参照）．

● べき乗剰余算処理の電力波形によって秘密かぎを得る

RSAのためのべき乗剰余算にはさまざまな実装法がありますが，基本的には指数e，あるいはdのビット・パターンに応じて乗剰余算と2乗剰余算を繰り返します．また，乗剰余算と2乗剰余算は加算と減算の繰り返しとして実装できます（あるいはモンゴメリ乗算と呼ばれる，除算をシフトで行う手法も用いられる）．

RSAのビット数は大きいため，1回の乗剰余算または2乗剰余算には専用ハードウェアでも数百～数千サイクルを要します．このときの電力波形を調べることで秘密かぎdを求めるのが，単純電力解析SPAです．

RSAを，指数ビットの左側から計算していく「左バイナリ法」の例を**図3**に示します．例えば，上述した暗号の例では秘密かぎが49でしたが，2進数で表すと110001となります．左バイナリ法では，これを左側から調べてビットが'0'ならば2乗剰余算を，'1'ならば2乗剰余算と乗剰余算（×29）を行います．オシロスコープで見ると，2乗算と乗算の処理は明らかに波形が異なるため，秘密かぎdが露呈するわけです．**図2**に示したRSAの音響攻撃も，このSPAと原理はまったく同じです．また，電磁波を利用した単純電磁波解析（SEMA：simple electromagnetic analysis）でも同じような攻撃が可能です．

なお，この左バイナリ法については，秘密かぎのビットが'1'のときだけ乗剰余算を行うことが問題なので，'0'のときにもダミーの乗算を入れるなどして比較的簡単にSPAの対策を行うことができます．

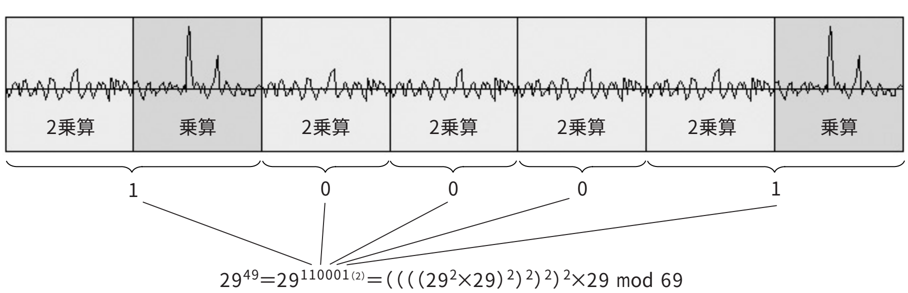


図3 左バイナリ法で実装したRSAに対するSPA
RSA暗号のべき乗剰余算は，2乗剰余算と乗剰余算を繰り返して実行される．左バイナリ法では，指数のビットが'1'のときだけ乗算が行われる．2乗剰余算，乗剰余算はそれぞれ数百～数千サイクルを要するので，このビット・パターンに応じた演算シーケンスはオシロスコープで簡単にとらえられる．

◉COLUMN 1◉ RSA暗号と素因数分解と離散対数問題

1,024ビットRSAのかぎを作るには，まず二つの512ビット前後の素数pとqを生成し，これを乗じて1,024ビットの合成数$n = p \times q$を作ります．このpとqを用いて拡張ユークリッド・アルゴリズムと呼ばれる方法でかぎのペアeとdを作ることができます．二つの素数を乗じて合成数nを作るのは簡単ですが，nからpとqを求める素因数分解は非常に難しい問題です．

米国RSA Security社は，素因数分解コンテストを継続して行っています（http://www.rsasecurity.com/rsalabs/node.asp?id=2092を参照）．現在の世界記録は2005年5月に発表された663ビット（10進で200けた）です．この素因数分解に要した演算量は，米国AMD社の64ビット・プロセッサ「Opteron（2.2GHz動作）」1チップで行った場合の55年分に相当するそうです．素因数分解は，合成数のビット数が長くなるにしたがって，指数関数的に難しくなります．

素因数分解ではなく，通信の第三者が次式の暗号文yと公開かぎeから平文xを直接求めることによってもRSA暗号を破ることができます（本文の式（1）を参照）．あるいは，暗号文の作成者が，本文の式（2）からxとyを満たす秘密かぎdを求めることも考えられます．

しかし，これらはいずれも素因数分解と同程度に非常に難しい問題で，離散対数問題と呼ばれています．

RSAの左バイナリ法では，秘密かぎのビット・パターンによって切り替わる演算が，全体の消費電力と処理時間を大きく左右しました．これに対して，共通かぎ暗号DESのハードウェア実装では，かぎのビット・パターンによらず，つねに16サイクルで処理できるため，1ビットのかぎの変化をSPAのような単純な電力波形でとらえることは極めて困難です．そこで次に，複数の波形を集めて統計処理を行う強力な差分電力解析DPAについて解説します．

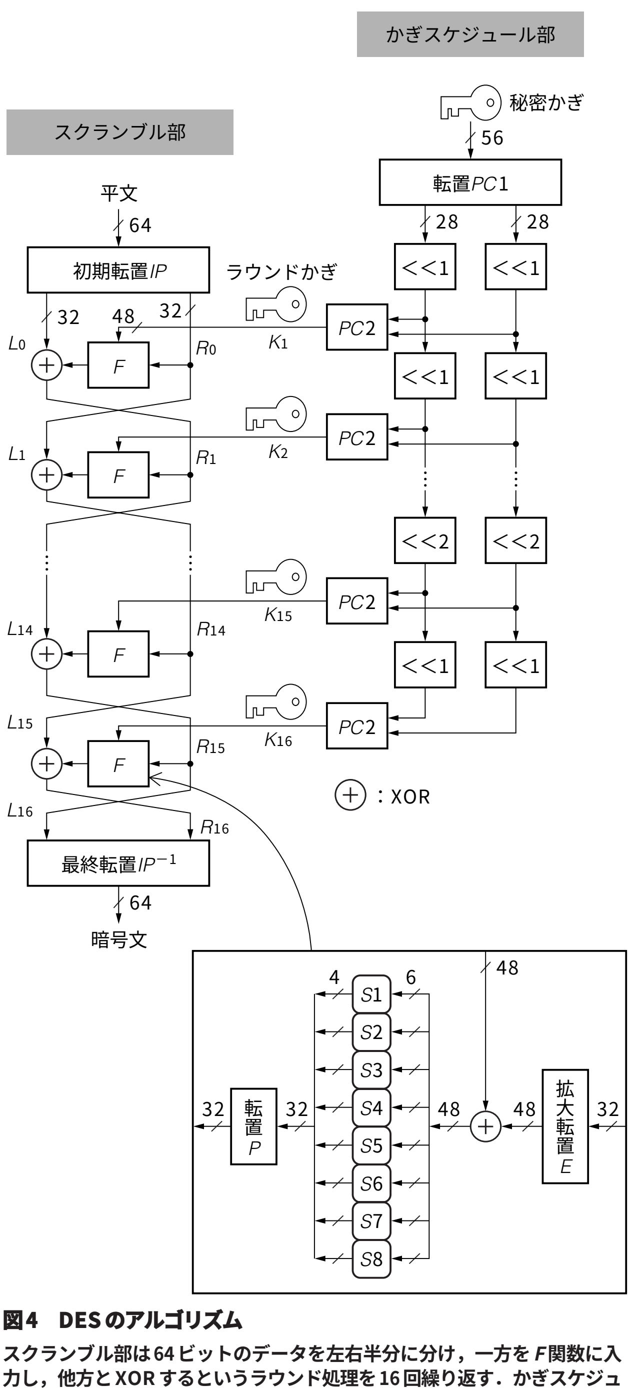


図4　DESのアルゴリズム

スクランブル部は64ビットのデータを左右半分に分け，一方をF関数に入力し，他方とXORするというラウンド処理を16回繰り返す．かぎスケジュール部は，56ビットの秘密かぎから48ビットのラウンドかぎを16個作り，各ラウンドのF関数に入力する．

2 DESをDPAで攻撃する

DESは共通かぎ暗号で，暗号化と復号のかぎは同じです．**図4**のように，56ビットのかぎを使い，64ビットのデータを左右32ビット（L_iとR_i）に分けて，32ビットのF関数による処理を16ラウンド回繰り返します．各ラウンドでは，48ビットのラウンドかぎK_iを用います．このK_iは，かぎスケジュール部において56ビットの秘密かぎをシフトし，ビット選択およびかくはん（$PC2$）することによって作られます．

暗号化と復号でスクランブル部の処理はまったく同じですが，かぎスケジュール部の順序が異なります．暗号化では$K_1 \to K_{16}$の順に，復号では$K_{16} \to K_1$の順に使用します．なお，DESのソフトウェアやハードウェアでパラメータとして与えるかぎは64ビットですが，このうち8ビットはパリティなので，暗号化や復号の処理に必要なのは残りの56ビットとなります．

暗号化は，まず初期転置IPによって平文の64ビットを並べ替え，次式のラウンド処理を16回繰り返し，最後にIPの逆変換である最終転置IP^{-1}を行います．

F関数の構成要素は次のとおりです．

- ビット重複を許して32ビットのデータを48ビットに並べ替える拡大転置E
- 48ビットかぎとのXOR（排他的論理和）
- S-Boxと呼ばれる，6ビット入力4ビット出力の8個のランダムな置換テーブル（$S1$～$S8$）
- 32ビット転置P

S-Box以外はすべて線形変換です．そのため，暗号の強さはこの非線形変換のS-Boxに大きく依存します．各S-Boxの変換規則は，**リスト1**に示すVerilog HDLのRTL記述を参照してください．

この規則がどうやって決められたのかは明らかにされておらず，米国政府には簡単に解読できるしくみが入っているのではないかという憶測もありました．そのため，新しい暗号の標準規格AES（Advanced Encryption Standard）で用いるS-Boxは，0～255の256個の整数で四則演算が閉じているガロア体$GF(2^8)$というものを用いて数学的に定義されています．こちらの回路実装については，参考文献

リスト1
S-Box

```
module SP(S, P);
  input  [1:48] S;◀── 48ビット→32ビット非線形変換のS-Boxと，32ビット転置Pを一つのモジュールとして記述
  output [1:32] P;

  reg [1:4] x1, x2, x3, x4, x5, x6, x7, x8;

  assign P = {
     x4[4], x2[3], x5[4], x6[1], x8[1], x3[4], x7[4], x5[1],
     x1[1], x4[3], x6[3], x7[2], x2[1], x5[2], x8[3], x3[2],
     x1[2], x2[4], x6[4], x4[2], x8[4], x7[3], x1[3], x3[1],
     x5[3], x4[1], x8[2], x2[2], x6[2], x3[3], x1[4], x7[1]
};

  always @(S[1:6]) begin
    case ({S[1], S[6], S[ 2: 5]})◀── S-Box S1．入力 S[1：6]のビット入れ替えに本質的な意味はないが，DESの仕様どおりに記述
      0: x1 = 14;   1: x1 =  4;   2: x1 = 13;   3: x1 =  1;
      4: x1 =  2;   5: x1 = 15;   6: x1 = 11;   7: x1 =  8;
      8: x1 =  3;   9: x1 = 10;  10: x1 =  6;  11: x1 = 12;
     12: x1 =  5;  13: x1 =  9;  14: x1 =  0;  15: x1 =  7;
     16: x1 =  0;  17: x1 = 15;  18: x1 =  7;  19: x1 =  4;
     20: x1 = 14;  21: x1 =  2;  22: x1 = 13;  23: x1 =  1;
     24: x1 = 10;  25: x1 =  6;  26: x1 = 12;  27: x1 = 11;
     28: x1 =  9;  29: x1 =  5;  30: x1 =  3;  31: x1 =  8;
     32: x1 =  4;  33: x1 =  1;  34: x1 = 14;  35: x1 =  8;
     36: x1 = 13;  37: x1 =  6;  38: x1 =  2;  39: x1 = 11;
     40: x1 = 15;  41: x1 = 12;  42: x1 =  9;  43: x1 =  7;
     44: x1 =  3;  45: x1 = 10;  46: x1 =  5;  47: x1 =  0;
     48: x1 = 15;  49: x1 = 12;  50: x1 =  8;  51: x1 =  2;
     52: x1 =  4;  53: x1 =  9;  54: x1 =  1;  55: x1 =  7;
     56: x1 =  5;  57: x1 = 11;  58: x1 =  3;  59: x1 = 14;
     60: x1 = 10;  61: x1 =  0;  62: x1 =  6;  63: x1 = 13;
    endcase
  end

  always @(S[7:12]) begin
    case ({S[7], S[12], S[ 8:11]})◀── S-Box S2～S8
       0: x2 = 15;   1: x2 =  1;   2: x2 =  8;   3: x2 = 14;
       4: x2 =  6;   5: x2 = 11;   6: x2 =  3;   7: x2 =  4;
                        ...中略...
      56: x8 = 15;  57: x8 = 12;  58: x8 =  9;  59: x8 =  0;
      60: x8 =  3;  61: x8 =  5;  62: x8 =  6;  63: x8 = 11;
    endcase
  end
endmodule
```

(3)で優れた設計が発表されているので，興味を持たれた方はそちらを参照してください．

DESのS-Boxは組み合わせ回路として実装されますが，転置変換はビット入れ替えのために配線をねじるだけなので，ドライバが入る場合を除いてゲート数は0です．データパスの残りは，32ビットXOR，48ビットXOR，64ビット・データ・レジスタ，56ビットかぎレジスタ，そしてセレクタといった簡単な部品で構成することができます．

● 何万もの波形を集めて解析することで秘密かぎを得る

DESの回路実装としては，一つのラウンド関数ブロックを繰り返し使用するループ・アーキテクチャが一般的です．このとき，暗号化の処理にはわずか16サイクルしかかかりません．また，かぎのビット・パターンの違いがRSAのように電力波形に明白に現れることもないため，SPAで攻撃することは困難です．

そこで，暗号化処理の電力波形を数千～数万パターン集めて統計処理することで，そこに含まれているわずかな秘密かぎの情報を増幅してやろうというのが差分電力解析DPAです．DPAは，まずかぎを予測し，そのかぎで得られるはずの中間データを計算します．そのうちの1ビット(選択関数)が'0'か'1'かで電力波形を$T0$と$T1$の2グループに分けて平均をとります．選択関数の1ビットに対応する回路の消費電力はわずかですが，かぎの予想が合っていて，数千～数万パターンを正しく振り分けることができれば，それを集めた波形にピークを見ることができます(**図5**)．予想が誤っているとランダムに分類された波形の平均をとることになるので，そのようなピークは現れません．

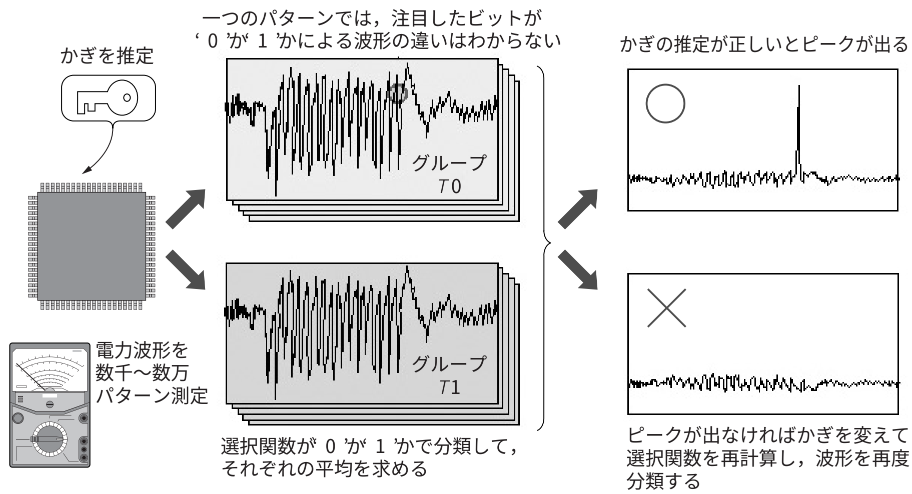


図5
DPAの概念図
かぎの違いによる中間データのビット変化に着目し，数千～数万の電力波形を分類してそれぞれの平均を求める．かぎの予測が合っていれば，平均化したグラフにピークが現れる．

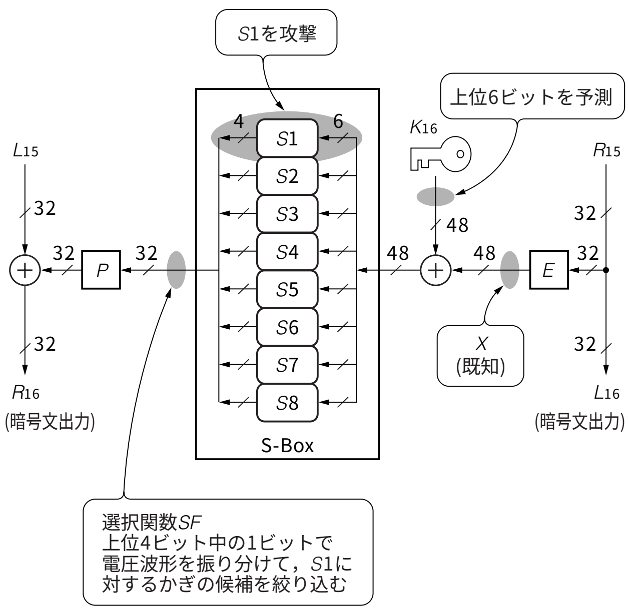


図6　DESの最終16ラウンドに対するDPA
48ビットのラウンドかぎを，S-Boxごとに$S1$から6ビットずつ決定していく．6ビットかぎを予測し，それに対応するS-Box出力のいずれか1ビットで電力波形を分類して平均する．

● DESの攻撃の手間は計測に要する2^9オーダ

それではDESのDPAをもう少し具体的に見ていくことにしましょう．選択関数として，さまざまなものを定義できますが，最終16ラウンドのS-Boxのbビット目の出力を利用した次式がよく使われます．

$$SF(X,\ K_{16},\ b) = Bit(SBox(X \oplus K_{16}),\ b) \quad \cdots\cdots(3)$$

図6からもわかるように，DESの最終16ラウンドはラウンド処理の入出力L_{16}とR_{16}が暗号文として外部から観測可能なので，解析にはもってこいです．1ラウンド目を解析に用いることもできますが，平文入力は未知であることが多いので，16ラウンドへの攻撃が一般的です．

また，はじめにラウンドかぎK_{16}を推定するわけですが，F関数入力の48ビット（実際には32ビットの拡大転置）すべてを順番に試していったのでは，組み合わせの数が膨大になります．これでは，総当たり攻撃とほとんど変わりありません．

そこでまず，F関数中の一つのS-Box $S1$の入出力関係を調べます．この入力6ビットは，ラウンドかぎK_{16}の上位6ビットと，拡大転置Eの出力Xの上位6ビットをXORしたものです．ここで，Xは暗号文出力L_{16}のビットを並べ替えただけなので攻撃者に既知です．そのため，$S1$に対する攻撃で予測するべきかぎのパターンは，K_{16}の上位6ビットに対する$2^6 = 64$パターンしかありません．そして，それに対する$S1$の出力は4ビットですが，そのうちのいずれか1ビットを選択関数SFとして，波形を$T0$と$T1$の2グループに分類するのです．それぞれのグループに属する波形の数を$N0$，$N1$とするとき，平均差分波形ΔTにピークが出れば攻撃成功です．

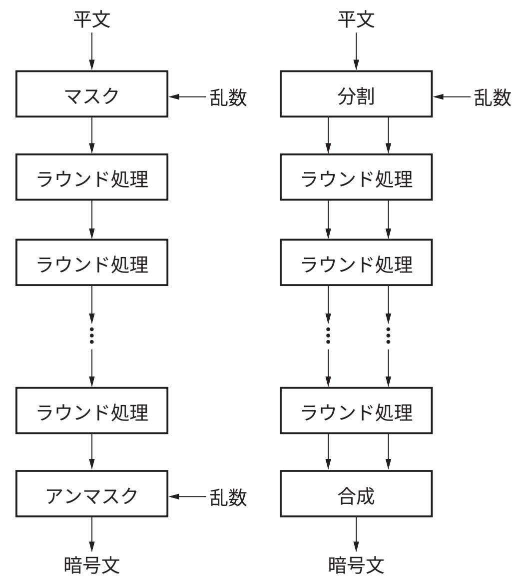
$$\Delta T = \frac{\sum T0}{N0} - \frac{\sum T1}{N1} \quad \cdots\cdots(4)$$

その後，$S2$～$S8$に対して順番に$S1$と同じような攻撃を行っていきます．ピークがうまく出ない場合，選択関数を変えたり，測定するサンプル数を増やしたり，可能であれば攻撃対象を1ラウンド目にしてみるといったことを行う必要があります．もちろん論理的な解析だけでなく，ノイ

ズを抑えたり，同期を正確に合わせたり，分解能を上げたりといった計測の精度向上のための努力も不可欠です．

なお，K_{16}とXのXORを選択関数とすれば，かぎを1ビットずつ簡単に決めていけるように思えます．これを行わない理由は，数ビットの入力に対して動く大きな論理回路であるS-Boxを通った後のビットを選択関数としたほうが，情報を電力差として取り出しやすいためです．

八つのS-Box（$S1$～$S8$）に対してうまくピークが出たならば，$2^6 \times 8 = 2^9$の手間で48ビットのかぎ情報が得られたことになります．残りの8ビットの情報は総当たりで調べても，$2^8 = 256$回の計算で求められます．この計算は一瞬で終わるため，DPAによるDESの攻撃の演算の手間は2^9オーダと言えるかもしれません（実際には演算よりも数千～数万波形を取得するほうにより多くの時間を要するが…）．これは，全数探索のオーダ2^{56}のわずか56兆分の1（$= 2^{-49}$）という驚くべき数字です．このように，DPAは非常に強力な攻撃法なのです．



(a) 乱数でマスク　**(b)** 乱数で分割

図7　DPA対策法の概念
乱数でデータをマスクしたり，処理を分割する．これにより，本来の中間データが現れないようにして，選択関数に波形の分類を誤らせる．

● 乱数を使って対策する

DPAの対策として，乱数を使って本来の中間データが現れないようにスクランブルをかける手法が提案されています．よく知られている対策としては，平文データを乱数でマスクしてからラウンド処理を行い，暗号文を出力するときに乱数マスクを外す方法（**図7(a)**）や，平文を乱数で二つのデータに分割し，ラウンド処理を二重化して最後に合成する方法（**図7(b)**）などがあります．

なお，**図7**はあくまでも概念を示したものです．このような対策は線形変換では容易なのですが，非線形変換である暗号アルゴリズムではそう簡単にはいきません．この部分を詳しく解説しているとかなり複雑になるため，ここでは割愛します．なお，本特集のAppendix（pp.113-114）でもっとも基本的なAkkarの手法による乱数マスクを用いたDPA対策について解説しています．簡略化したサンプル・コードも公開されているので（http://www.aoki.ecei.tohoku.ac.jp/crypto/からダウンロードできる），興味のある方はフル実装にもチャレンジしてみてください．

また，このようなアルゴリズムによる対策だけでなく，トランジスタ・レベルの防御やその実装評価も行われています（p.112のコラム「トランジスタ・レベルのサイドチャネル攻撃対策」を参照）．これらの対策法をさらに詳しく知りたい方は，暗号ハードウェアとシステム実装に関する国際会議CHES（Cryptrographic Hardware and Embedded Systems）[5]を参照されるとよいでしょう．ここ数年はサイドチャネル攻撃に関する論文が半数を占めていることからも，この研究分野が今いかに注目されているかがわかると思います．

2

参考・引用*文献

(1) A. Shamir and E. Trome ; Acoustic cryptanalysis - On nosy people and noisy machines, http://www.wisdom.weizmann.ac.il/~tromer/acoustic/
(2) P. Kocher ; Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems, LNCS1109, pp.104-113, 1996（http://www.cryptography.com/resources/whitepapers/TimingAttacks.pdf）．
(3) Design Wave設計コンテスト2004に関するWebサイト，http://www.cqpub.co.jp/dwm/contest/2004/youkou.htm
(4) P.Kocher, J. Jaffe, and B. Jun ; Differential Power Analysis, CRYPTO '99, LNCS1666, pp.388-397, 1999（http://www.cryptography.com/resources/whitepapers/DPA.pdf）．
(5) CHESのWebサイト，http://islab.oregonstate.edu/ches/

さとう・あかし
日本アイ・ビー・エム（株）東京基礎研究所

◉COLUMN 2◉ トランジスタ・レベルのサイドチャネル攻撃対策

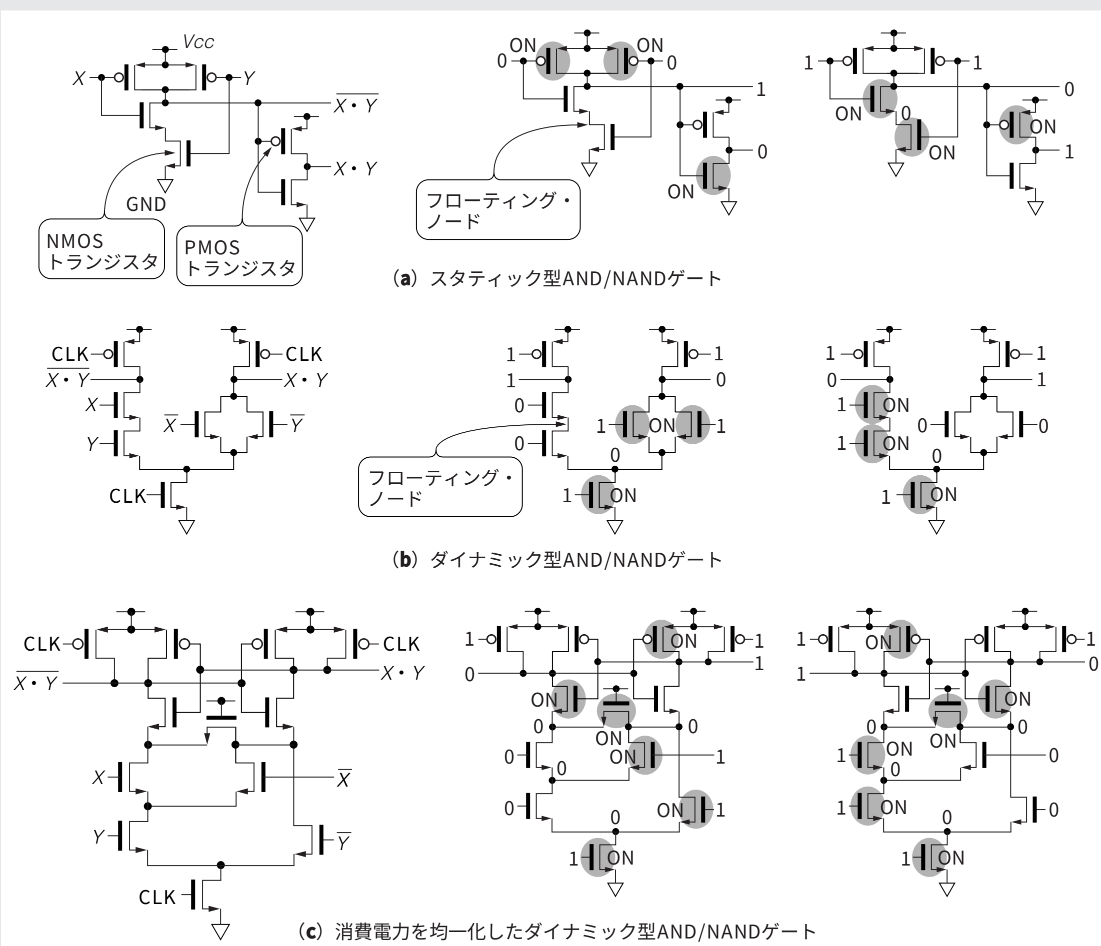
SPAやDPAは，演算に依存した消費電力の違いを利用する攻撃法です．そこで，回路のくふうにより，消費電力がデータに依存せず，つねに一定となるような対策法が提案されています．

ASICライブラリの論理ゲートの大半は，相補的に動作するPMOSトランジスタとNMOSトランジスタの組を使うスタティック型のCMOS論理を使っています．例として，**図A-1(a)**にスタティック型のAND/NANDゲートのトランジスタ構成を示します．PMOSトランジスタはゲートが‘0’(GNDレベル)のとき‘1’を通し，NMOSはゲートが‘1’(V_{cc}レベル)のとき‘0’を通します．ANDとNANDのロジックは，2入力が[0，0]，[0，1]，[1，0]の3パターンで変化しても出力はそれぞれ‘0’と‘1’に固定です．入力が[1，1]のときだけ，出力がANDでは‘1’に，NANDでは‘0’になります．したがって，消費電力は入力データの変化に大きく依存します．

そこで，どのような入力であっても二つの出力のどちらか一方が変化する，2線式のダイナミック型ゲートが提案されています．

図A-1(b)のダイナミック型ゲートは2入力[X, Y]に加えて，その反転した信号[$\overline{X}$, $\overline{Y}$]を入力としてAND出力$\overline{X\cdot Y}$とNAND出力$X\cdot Y$を生成します．クロック信号(CLK)が‘0’のとき，電源V_{cc}につながる二つのPMOSトランジスタがONとなり，二つの出力線はいずれも‘1’にチャージ・アップされます．次に，CLK＝1になると，V_{cc}側のPMOSがOFF，GND側のNMOSがONとなり，2組の入力に応じて出力線のいずれかが‘0’に引き抜かれます．しかし，これでは電力攻撃への対策としてはまだ不十分です．**図A-1(b)**の中央と右の状態ではON状態にあるトランジスタの接続関係が対称ではなく，電荷が引き抜かれる拡散層(トランジスタのソースとドレイン)容量が異なるため，入力のパターンによってやはり消費電力が異なってしまうのです．

そこで，**図A-1(c)**に示すような回路が提案されています．図から演算時に出力線の下側のすべてのノードは‘0’に引き抜かれ，またONしているトランジスタも左右対称であることがわかります．この回路はSABL(sense amplifier based logic)とも呼ばれています．その名は，DRAMやSRAMなどのメモリ・セルから出力される微小な信号の増幅などに広く用いられる差動増幅器を論理回路に応用したことに由来します．入力信号のパターンによらず消費電力が一定で，かつ高速動作が可能なのですが，その反面，多くのトランジスタと大きな消費電力を要することが課題であると言えます．



図A-1 ゲート・レベルの対策